dotnet new webapi -n Restaurants.API --no-openapi -controllers

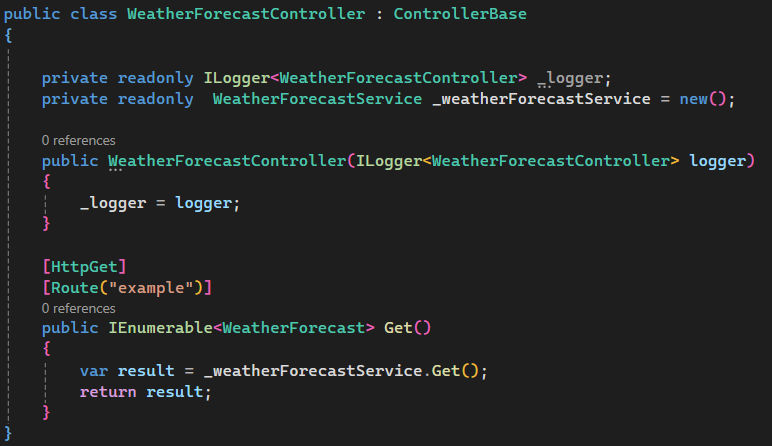
dotnet new sln

dotnet new sln -n Restaurants ---> to custom name

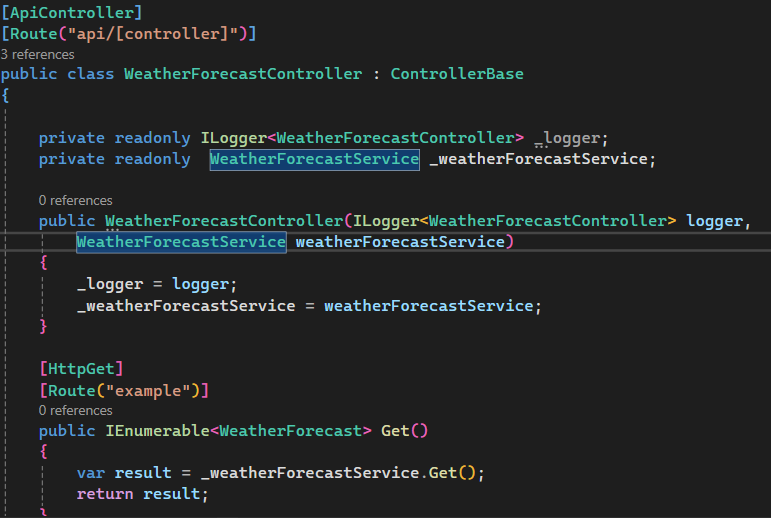
dotnet sln add ./Restaurants.API--->connecting project to solution file

<https://www.youtube.com/watch?v=E6sUJWwZLwE>

Creating and using Services

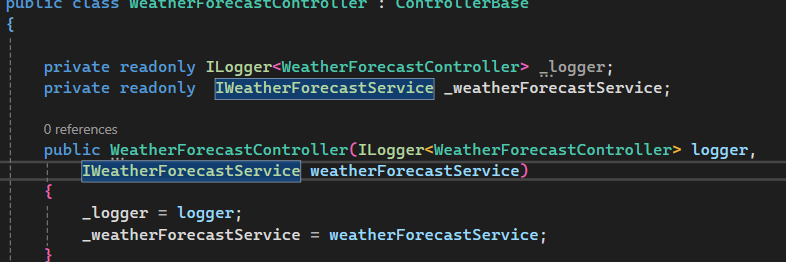


Or



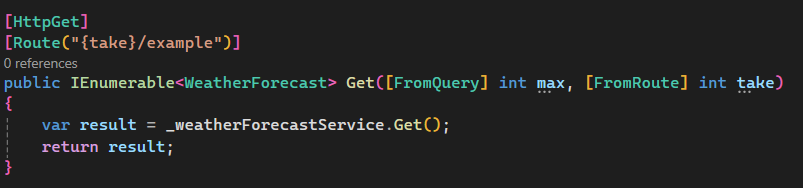
And add in program.cs 🡺 builder.Services.AddScoped<WeatherForecastService>();



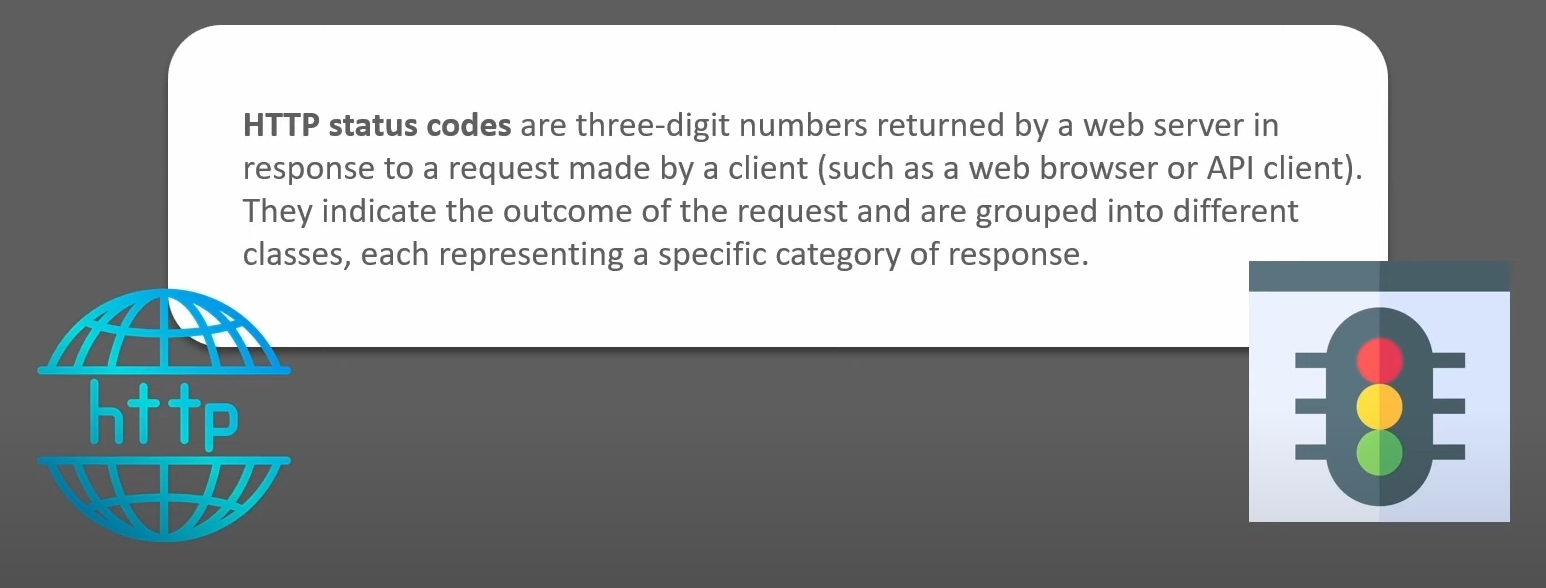
Same Service class but using interface  
  
  


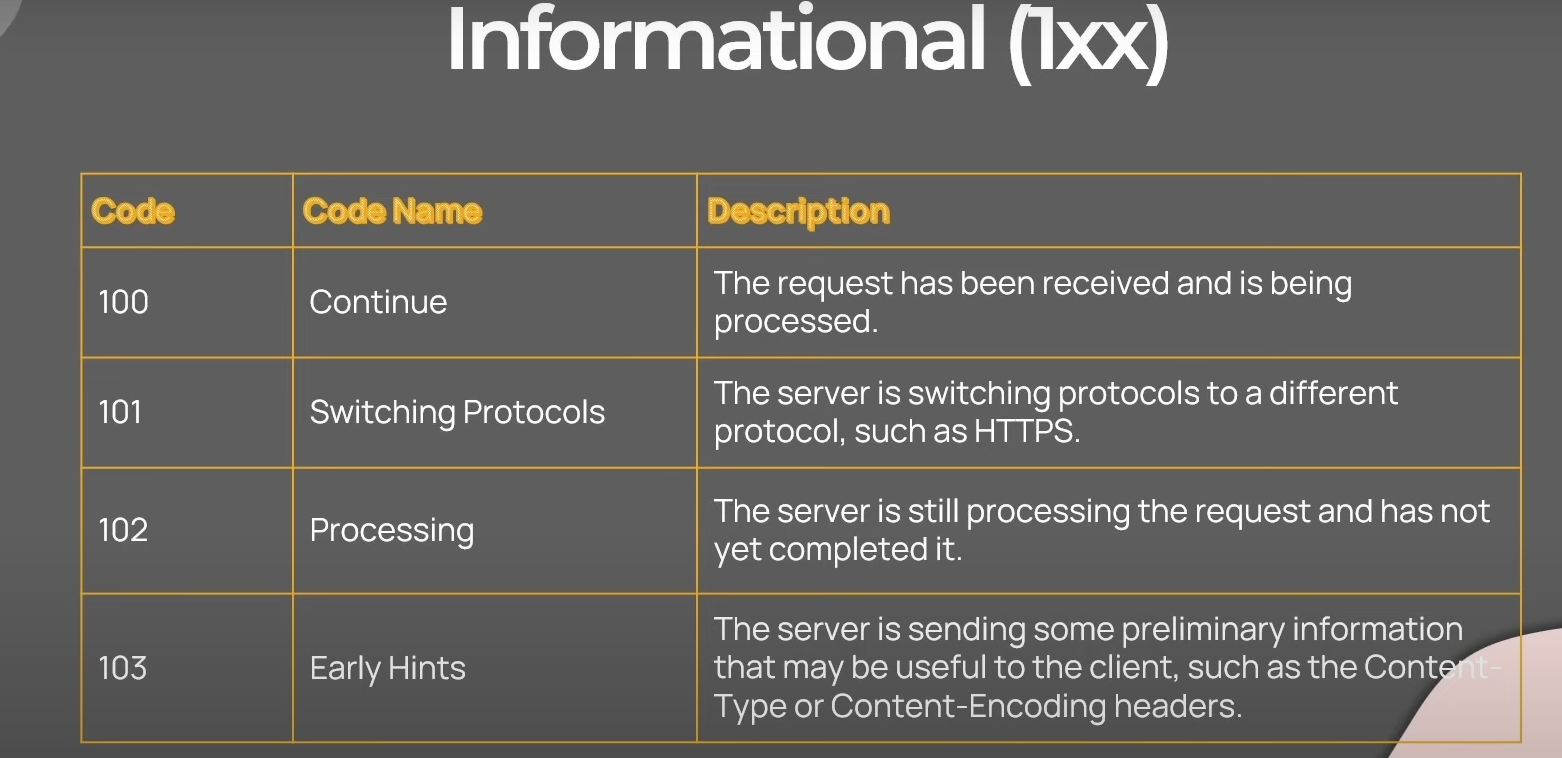
Program.cs  

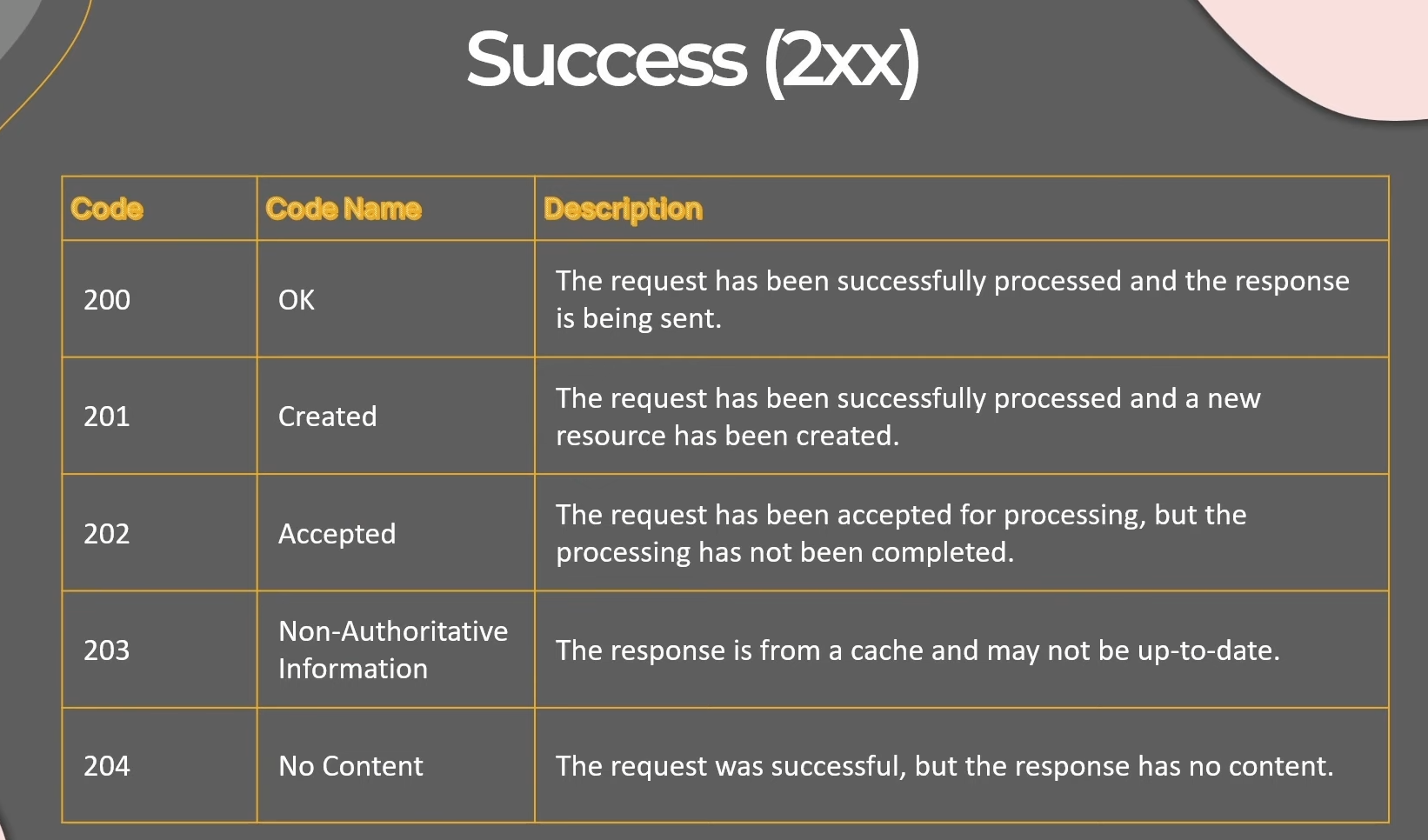

Adding Route and showing different Request Type (query, path and body)

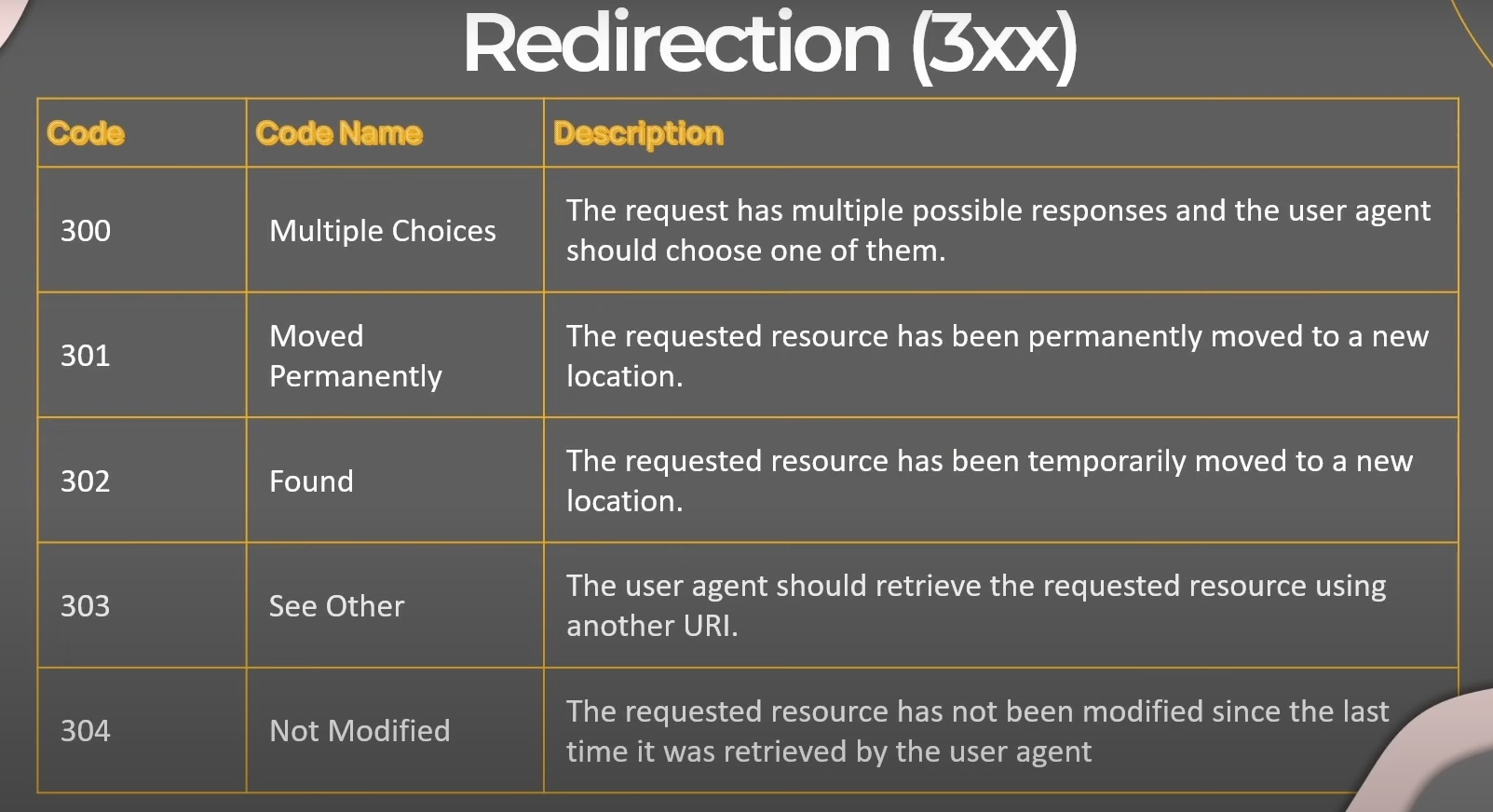


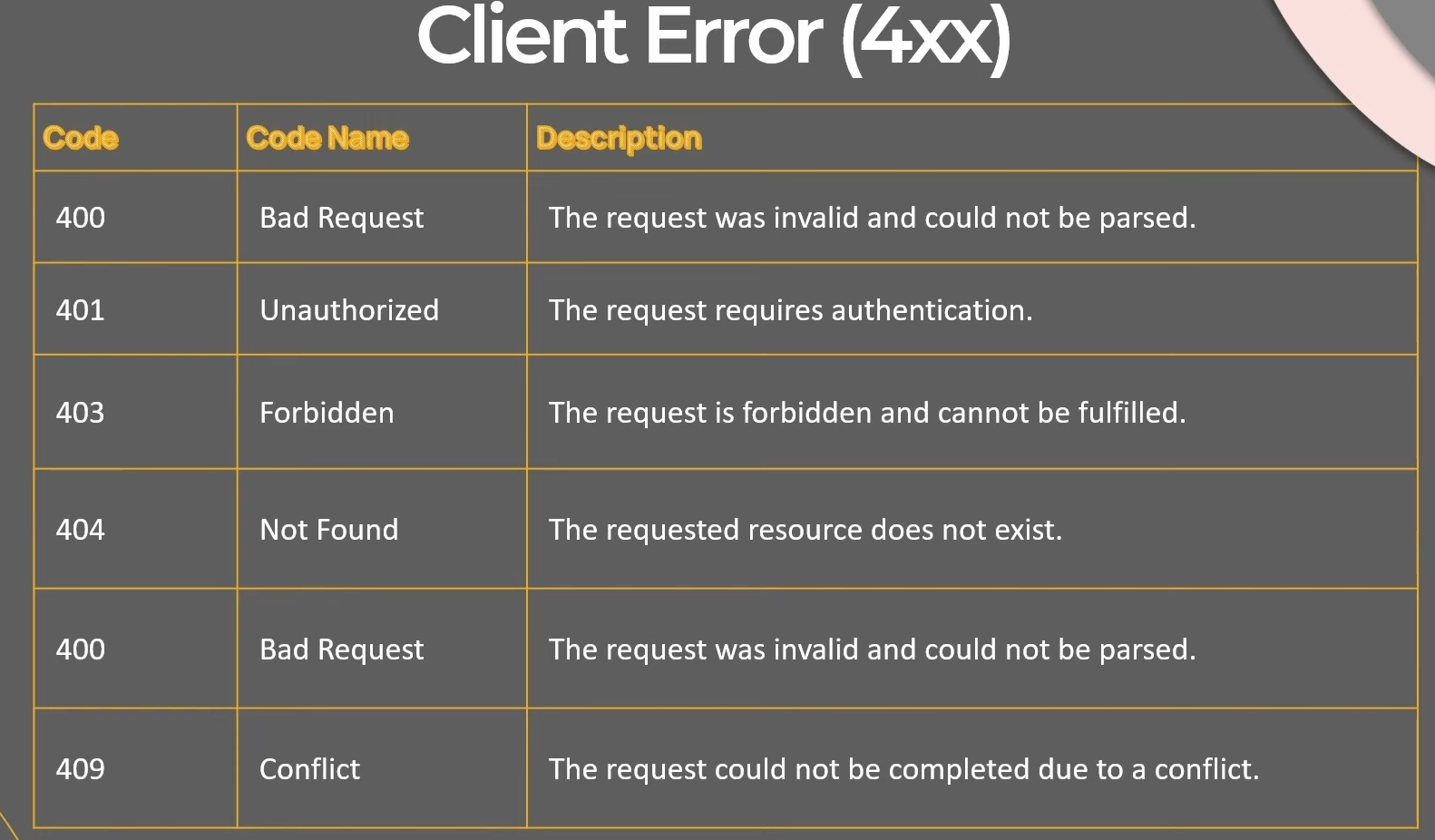


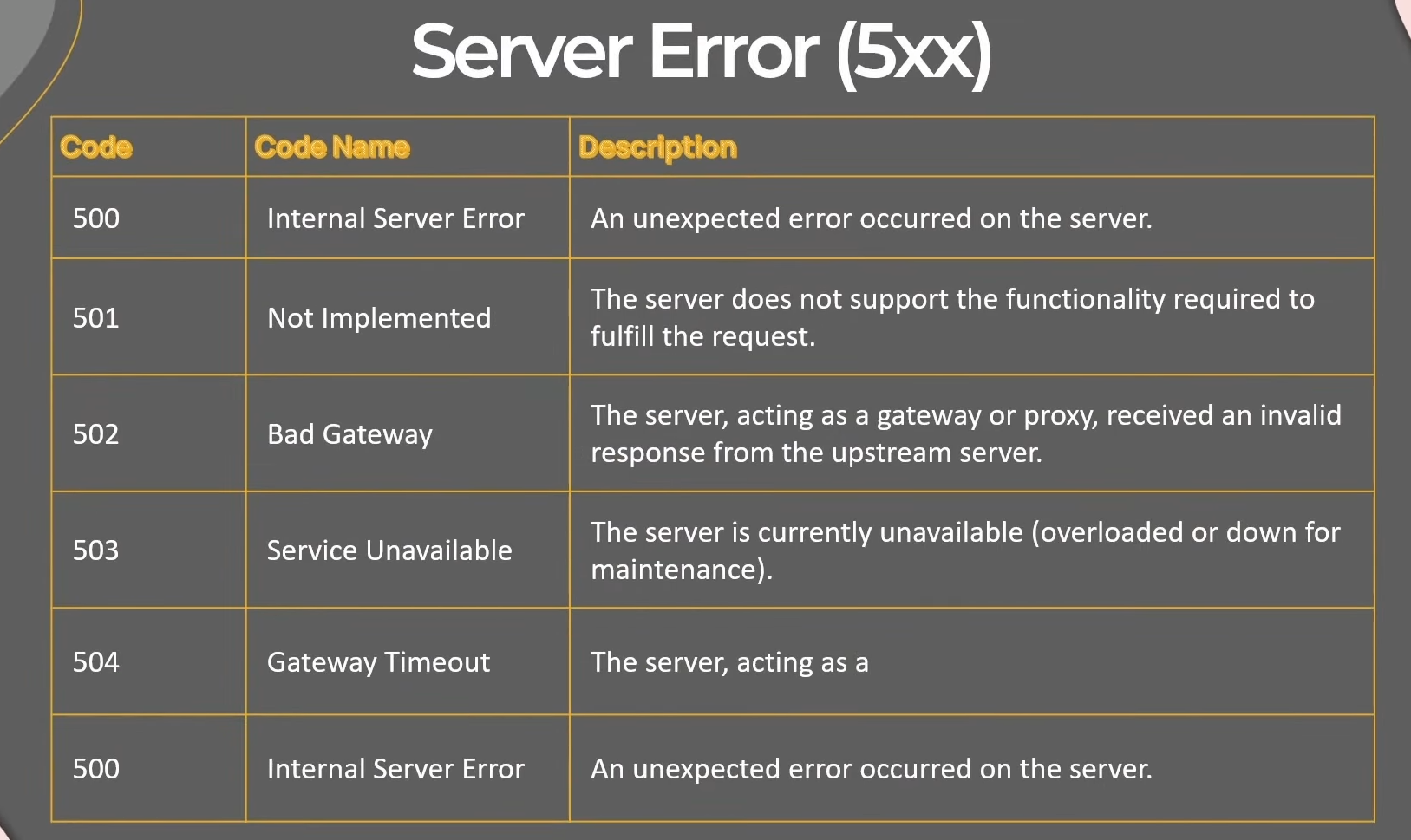


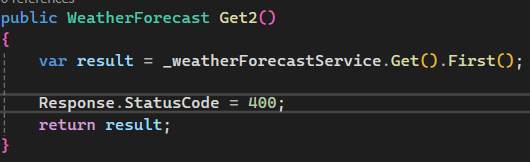
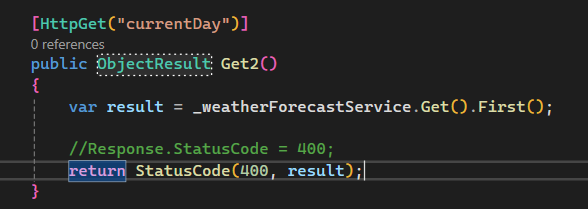




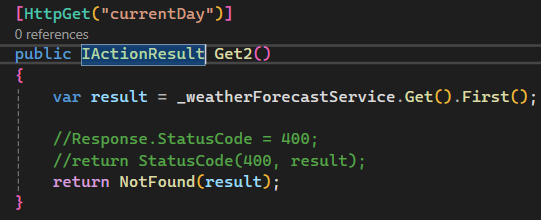


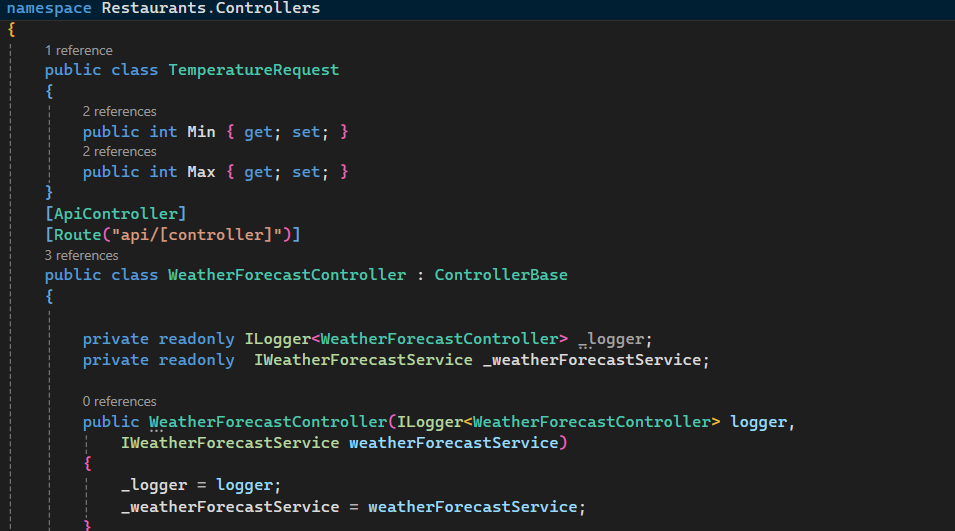




Changing Status Code in controller Response  
 or 

return BadRequest(result); return Ok(result); return Created(result); return NotFound(result);



Sample Controller  


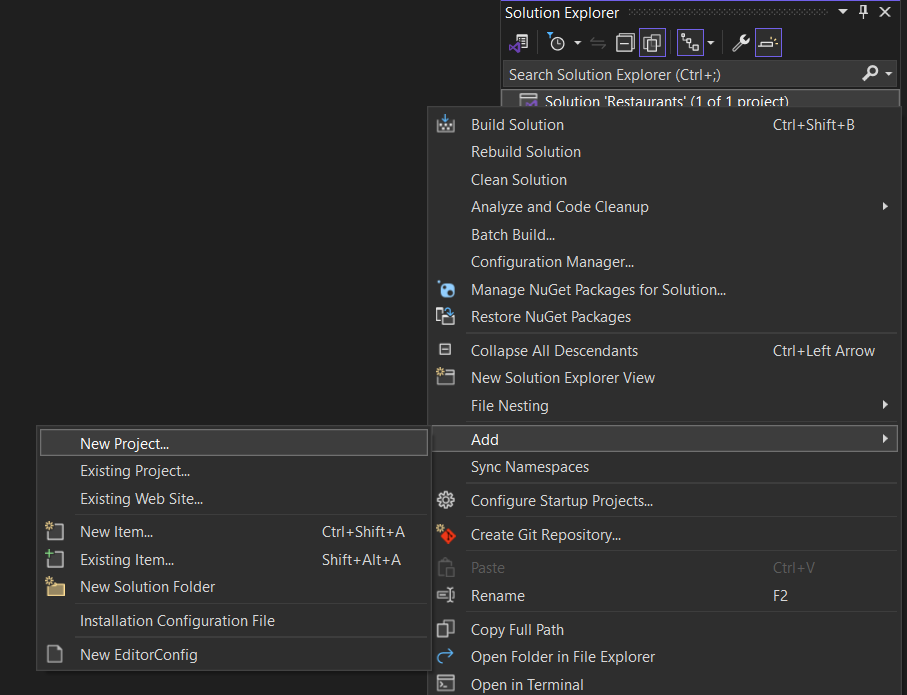


Clean Architecture  
-Domain

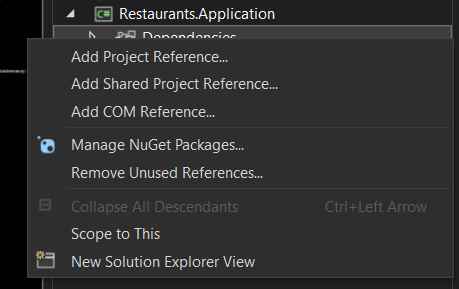
-Application

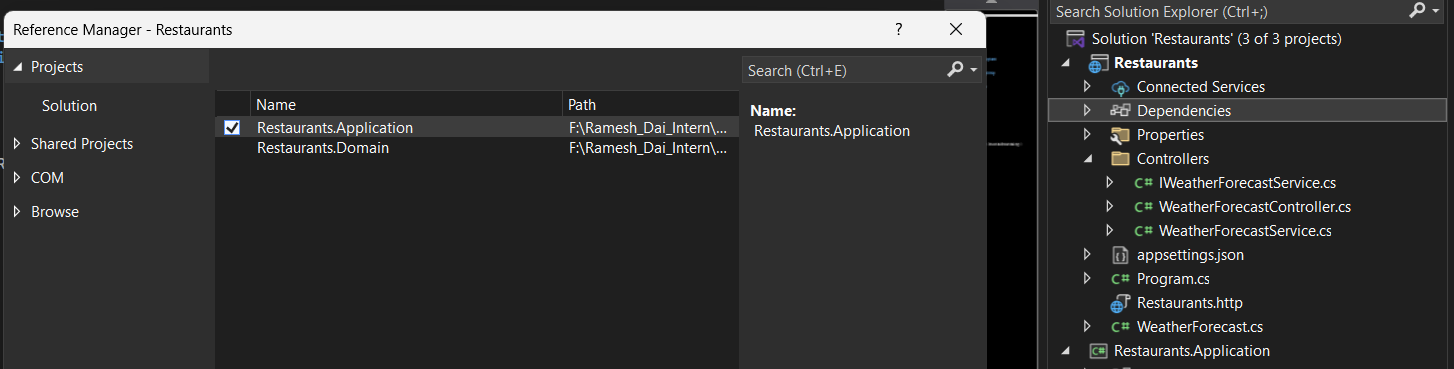
-Presentation and Infrastructure

Add Domain Project to Solution

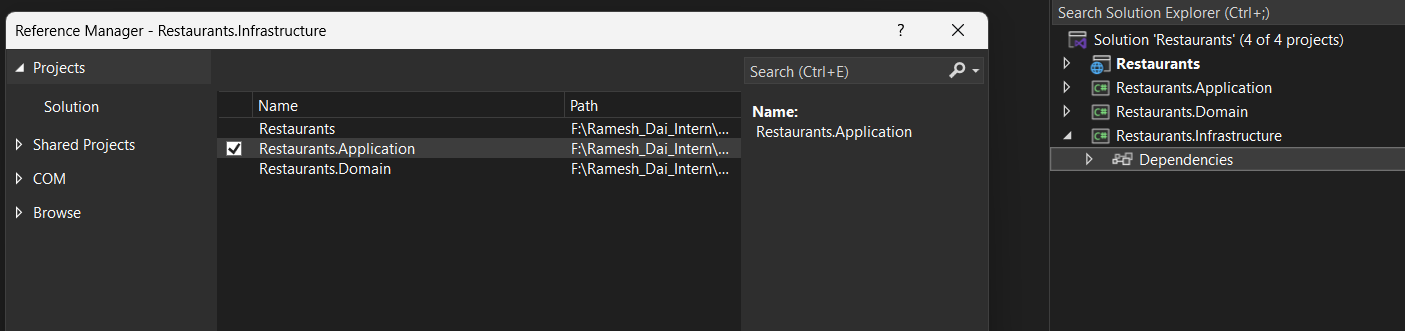
  
  
Creater Restaurants.Domain as Class Library

Create Restaurants.Application as Class Library

* Add reference of Domain for Application Dependancy
  + 
* Add reference of Application for Main Restaurants API dependencies



* Add reference of Application for Infrastructure dependencies

  
  
Shift + F2 to create new Folder and new Class

If want to create folder add slash at the end without extension cs

Creating First Entities

namespace Restaurants.Domain.Entities

{

public class Restaurant

{

public int Id { get; set; }

public string Name { get; set; } = default!;

public string Description { get; set; } = default!;

public string Category { get; set; } = default!;

public bool HasDelivery { get; set; }

public string? ContactEmail { get; set; }

public string? ContactNumber { get; set; }

public Address? Address { get; set; }

public List<Dish> Dishes { get; set; } = new();

}

}namespace Restaurants.Domain.Entities

{

public class Address

{

public string? City { get; set; }

public string? Street { get; set; }

public string? PostalCode { get; set; }

}

}namespace Restaurants.Domain.Entities

{

public class Dish

{

public int Id { get; set; }

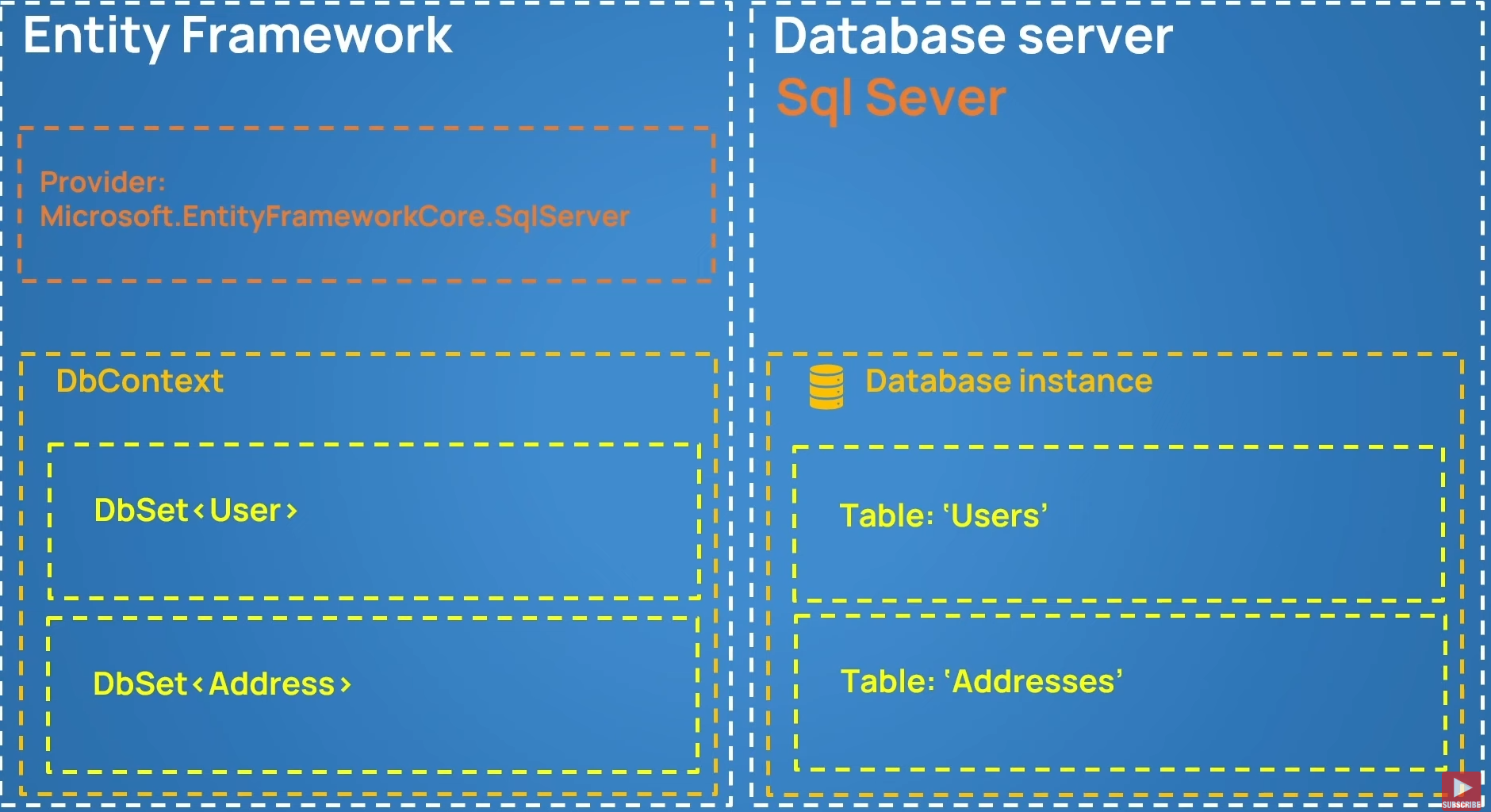
public string? Name { get; set; } = default!;

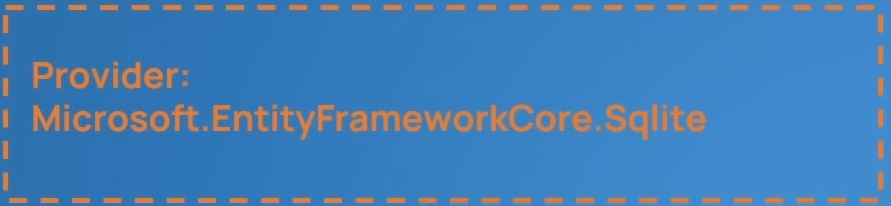
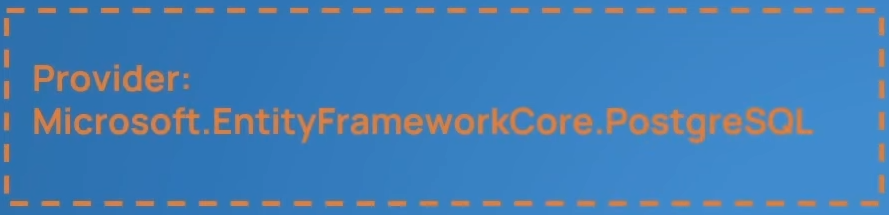
public string? Description { get; set; } = default!;

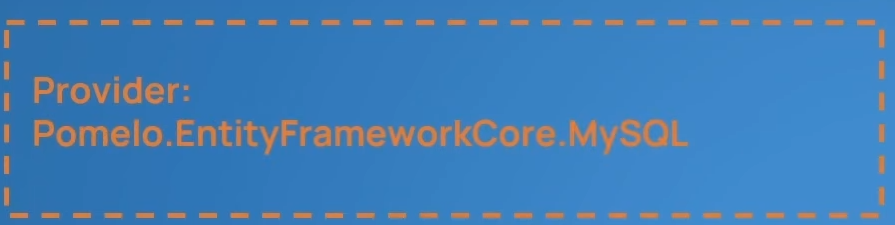
public decimal Price { get; set; }

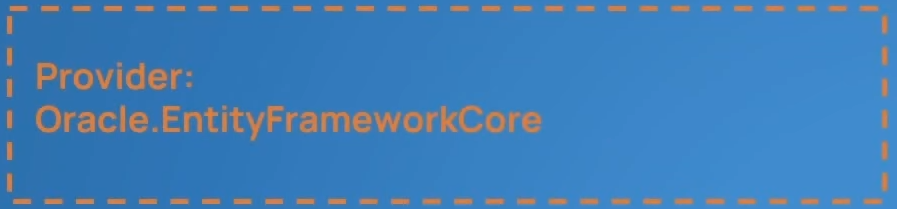
}

}







Creating Dbcontext and Registering Tables  
using Microsoft.EntityFrameworkCore;

using Restaurants.Domain.Entities;

namespace Restaurants.Infrastructure.Persistence

{

internal class RestaurantsDbContext : DbContext

{

internal DbSet<Restaurant> Restaurants { get; set; }

internal DbSet<Dish> Dishes { get; set; }

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

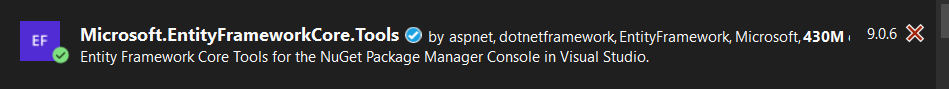
{

optionsBuilder.UseSqlServer("Server=DESKTOP-DMBAGSB\\SQLEXPRESS;Database=RestaurantsDb;Trusted\_Connection=True");

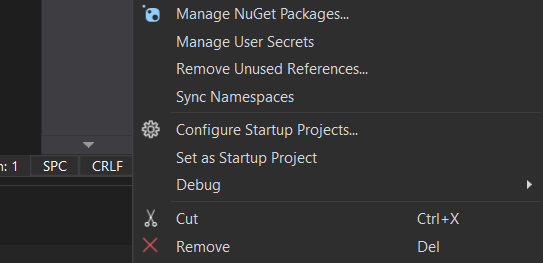
}

}

}



Add nugget package in infrastrucuture to perform migrations

Set Restaurants.Infrastructure as startup project  
  
go to package manager console

And type add-migration intit

Adding Addresstable inside the restaurant

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

base.OnModelCreating(modelBuilder);

modelBuilder.Entity<Restaurant>()

.OwnsOne(r => r.Address);

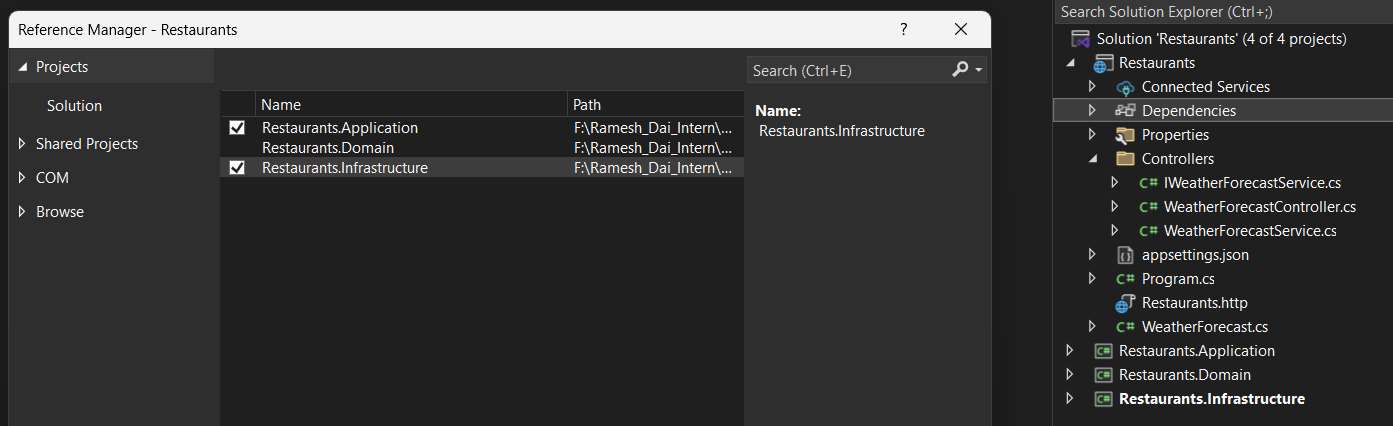
modelBuilder.Entity<Restaurant>()

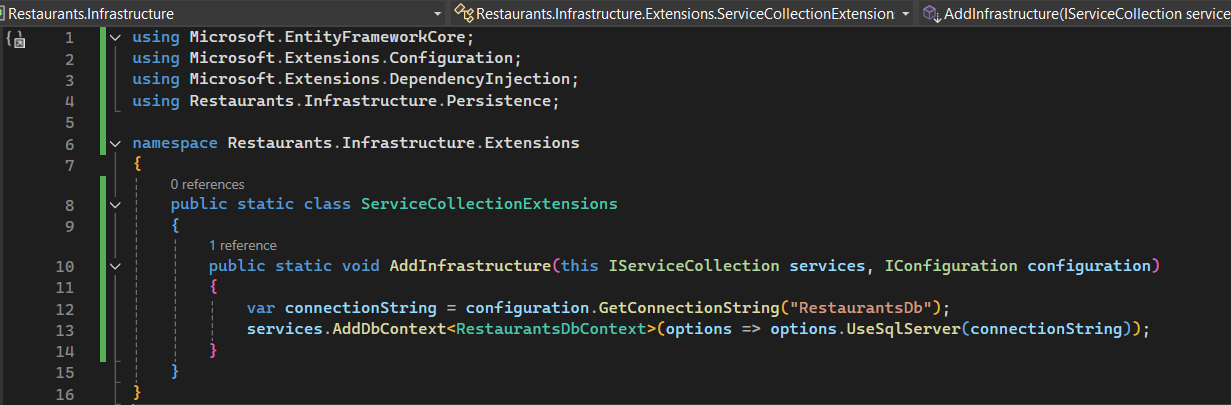
.HasMany(r => r.Dishes)

.WithOne()

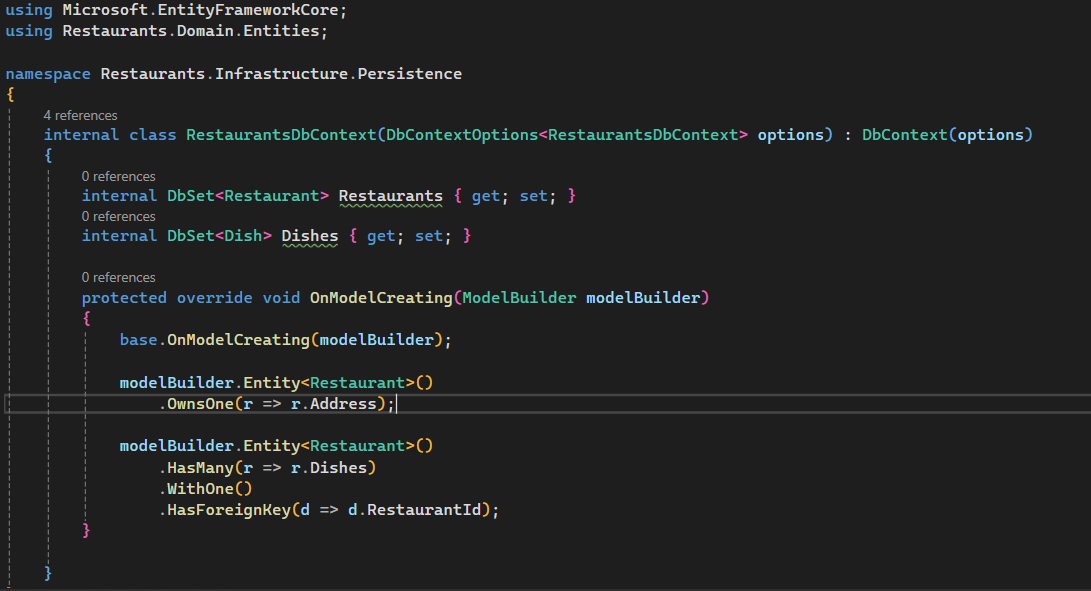
.HasForeignKey(d => d.RestaurantId);

}

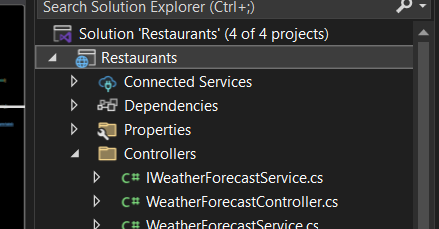
Connecting Infrastructure layer to main presentation layer  
For connecting database   


Restaurants.Infrastructure.Extensions/ServiceCollectionExtensions.cs  


RestaurantDbContext.cs



set Restaurants.API as startupProject.

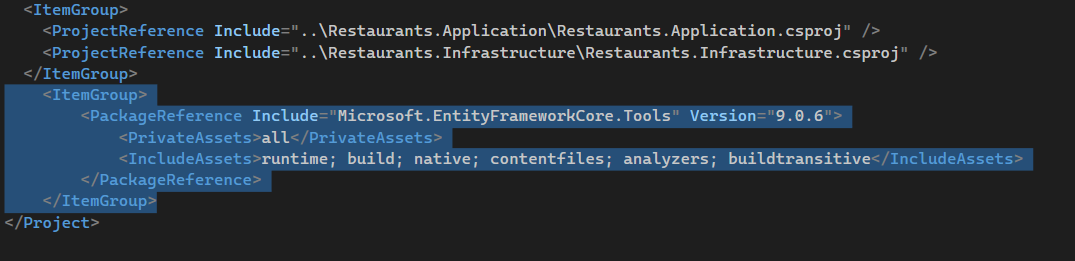
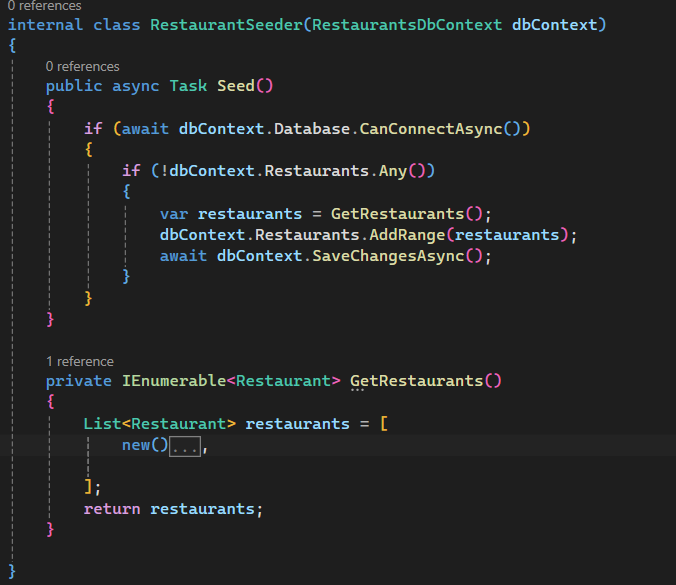
  
  
 <ItemGroup>

<PackageReference Include="Microsoft.EntityFrameworkCore.Tools" Version="9.0.6">

<PrivateAssets>all</PrivateAssets>

<IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>

</PackageReference>

</ItemGroup>  
  
Add this to Restaurants to API to be able to update database when restaurants.API is setup as startup Project  
  
  
Seeding the Database  


List<Restaurant> restaurants = [

new()

{

Name = "KFC",

Category = "Fast Food",

Description = "KFC is an AMerican food Company",

ContactEmail = "caont@yahoo.com",

HasDelivery = true,

Dishes =

[

new()

{

Name = "Nashville Hot Chicken",

Description = "Nash (10pcs)",

Price = 10.30M,

},

new()

{

Name = "Chicken Nuggets",

Description = "Nuggets (5pcs)",

Price = 5.30M,

},

],

Address = new()

{

City = "London",

Street = "Cork St 5",

PostalCode = "WC2N 5DU",

}

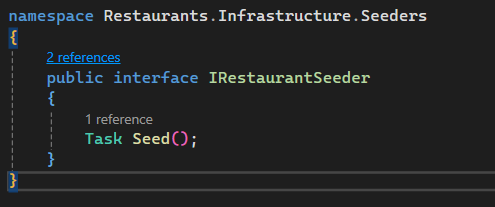
},

];

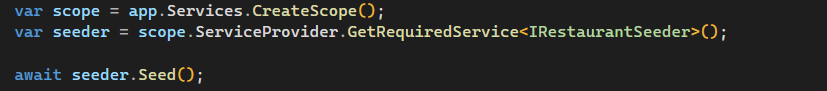
Also change this



Change this



Program.cs



### Chat GPT Code Step-by-Step Plan

**🔧 Step 1: Install Required NuGet Packages**

Run this from the Restaurants (API) root using the terminal:

dotnet add Restaurants.Application package Microsoft.Extensions.DependencyInjection.Abstractions

dotnet add Restaurants.Infrastructure package Microsoft.EntityFrameworkCore

dotnet add Restaurants.Infrastructure package Microsoft.EntityFrameworkCore.SqlServer

dotnet add Restaurants.Infrastructure package Microsoft.EntityFrameworkCore.Tools

dotnet add Restaurants.Infrastructure package Microsoft.AspNetCore.Authentication.JwtBearer

dotnet add Restaurants package Microsoft.AspNetCore.Authentication.JwtBearer

**📁 Folder/Project Layout Reminder**

* Restaurants.Domain: Entities and interfaces
* Restaurants.Application: DTOs, interfaces, services
* Restaurants.Infrastructure: EF Core, services implementation
* Restaurants: API, Controllers, JWT setup

**✅ Step 2: Add Dish DTOs and Interfaces**

**📄 Application/DTOs/DishDto.cs**

namespace Restaurants.Application.DTOs;

public class DishDto

{

public int Id { get; set; }

public string? Name { get; set; }

public string? Description { get; set; }

public decimal Price { get; set; }

}

**✅ Step 3: Create Dish Service Interface**

**📄 Application/Interfaces/IDishService.cs**

using Restaurants.Application.DTOs;

namespace Restaurants.Application.Interfaces;

public interface IDishService

{

Task<IEnumerable<DishDto>> GetAllAsync();

}

**✅ Step 4: Implement Dish Service**

**📄 Infrastructure/Services/DishService.cs**

using Microsoft.EntityFrameworkCore;

using Restaurants.Application.DTOs;

using Restaurants.Application.Interfaces;

using Restaurants.Infrastructure.Persistence;

namespace Restaurants.Infrastructure.Services;

internal class DishService(RestaurantsDbContext dbContext) : IDishService

{

public async Task<IEnumerable<DishDto>> GetAllAsync()

{

return await dbContext.Dishes

.Select(d => new DishDto

{

Id = d.Id,

Name = d.Name,

Description = d.Description,

Price = d.Price

})

.ToListAsync();

}

}

**✅ Step 5: Register DishService in DI**

**📄 Infrastructure/Extensions/ServiceCollectionExtensions.cs**

Add this line:

services.AddScoped<IDishService, DishService>();

Your method becomes:

public static void AddInfrastructure(this IServiceCollection services, IConfiguration configuration)

{

var connectionString = configuration.GetConnectionString("RestaurantsDb");

services.AddDbContext<RestaurantsDbContext>(options => options.UseSqlServer(connectionString));

services.AddScoped<IRestaurantSeeder, RestaurantSeeder>();

services.AddScoped<IDishService, DishService>();

}

**✅ Step 6: Create Controller**

**📄 Restaurants/Controllers/DishesController.cs**

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Restaurants.Application.Interfaces;

namespace Restaurants.Controllers;

[ApiController]

[Route("api/[controller]")]

[Authorize]

public class DishesController(IDishService dishService) : ControllerBase

{

[HttpGet]

public async Task<IActionResult> Get()

{

var dishes = await dishService.GetAllAsync();

return Ok(dishes);

}

}

**✅ Step 7: JWT Authentication Setup**

**📄 Restaurants/appsettings.json**

"JwtSettings": {

"Issuer": "YourAppIssuer",

"Audience": "YourAppAudience",

"Key": "superSecretKey@345"

}

**📄 Restaurants/Program.cs**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

using Restaurants.Infrastructure.Extensions;

var builder = WebApplication.CreateBuilder(args);

// Add services

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

builder.Services.AddInfrastructure(builder.Configuration);

// JWT Auth

var jwtSettings = builder.Configuration.GetSection("JwtSettings");

builder.Services

.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)

.AddJwtBearer(options =>

{

options.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = jwtSettings["Issuer"],

ValidAudience = jwtSettings["Audience"],

IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(jwtSettings["Key"]!))

};

});

var app = builder.Build();

app.UseSwagger();

app.UseSwaggerUI();

app.UseAuthentication(); // 🔐 Important

app.UseAuthorization();

app.MapControllers();

app.Run();

**✅ Step 8: Create a Token Generator (for testing login)**

You can mock login now.

**📄 Restaurants/Controllers/AuthController.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

namespace Restaurants.Controllers;

[ApiController]

[Route("api/[controller]")]

public class AuthController(IConfiguration configuration) : ControllerBase

{

[HttpPost("login")]

public IActionResult Login([FromBody] LoginRequest request)

{

// Hardcoded for testing

if (request.Username != "admin" || request.Password != "password")

return Unauthorized();

var claims = new[]

{

new Claim(ClaimTypes.NameIdentifier, request.Username),

new Claim(ClaimTypes.Role, "Admin")

};

var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(configuration["JwtSettings:Key"]!));

var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);

var token = new JwtSecurityToken(

issuer: configuration["JwtSettings:Issuer"],

audience: configuration["JwtSettings:Audience"],

claims: claims,

expires: DateTime.Now.AddHours(1),

signingCredentials: creds

);

return Ok(new { token = new JwtSecurityTokenHandler().WriteToken(token) });

}

}

public record LoginRequest(string Username, string Password);

**✅ Step 9: Use the API**

1. **Start the API**.
2. **POST** to /api/auth/login with:

{

"username": "admin",

"password": "password"

}

Get the token from response.

1. **Call** GET /api/dishes with:

Authorization: Bearer <your\_token>